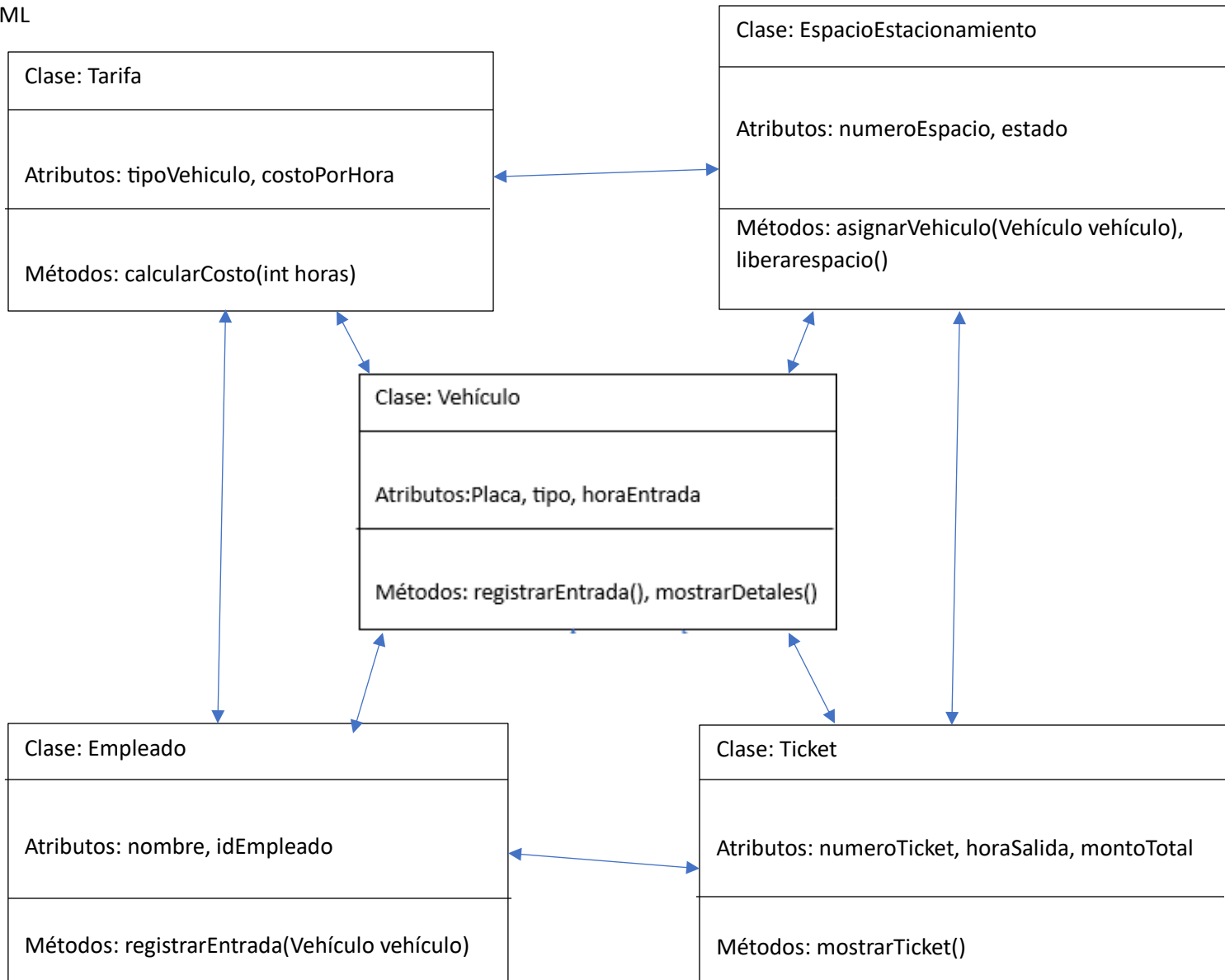


Actividad de Aprendizaje Contacto Docente N.º 2 Primer Parcial

UML



Programación Orientada a Objetos (POO).

Nombre: Nicolás Aldair Gualoto Piñajo

NRC: 1323

Tema de la actividad: Creación de objetos y UML

Tipo de actividad: Diseñado y Modelado.

Descripción de la Actividad:

Diseñe 5 objetos diferentes con su correspondiente diagrama UML, asegurándose de mostrar las relaciones entre ellos.

Sistema de parqueadero

Vehículo Representa los vehículos que ingresan al parqueadero, con atributos como placa, tipo y hora de entrada. Se relacionaria con cada Espacio de Estacionamiento por asociación.

Espacio de Estacionamiento: Administra los espacios ticket disponibles con atributos como un número espacio y podemos ponerle el estado de (ocupado o libre). Se relaciona con Vehículo por asociación.

Tarifa: Dara el costo del estacionamiento, con atributos como tipo de vehículo y costo por hora. Se relaciona con ticket por asociación.

Empleado: Gestiona las entradas y salidas de los vehículos, con atributos como nombre e id. Se relaciona con vehículo por asociación.

Ticket: Registra la información del pago, con atributos como número ticket, hora salida y monto total. Se relaciona con vehículo y tarifa por asociación.

Código:

```
class Vehículo {
    String placa;
    String tipo;
    String horaEntrada;

    public Vehículo (String placa, String tipo, String horaEntrada) {
        this.placa = placa;
        this.tipo = tipo;
        this.horaEntrada = horaEntrada;
    }

    public void registrarEntrada () {
        System.out.println("Vehículo con placa " + placa + " registrado a las " + horaEntrada);
    }

    public void mostrarDetalles () {
        System.out.println("Vehículo con placa " + placa + " registrado a las " + horaEntrada);
    }
}

class EspacioEstacionamiento {
    int numeroEspacio;
    boolean estado;

    public EspacioEstacionamiento (int numeroEspacio) {
        this.numeroEspacio = numeroEspacio;
        this.estado = false;
    }

    public void asignarVehículo (Vehículo vehículo) {
        if (!estado) {
            estado = true;
            System.out.println("Espacio " + numeroEspacio + " asignado al vehículo con placa " + vehículo.placa);
        } else {
            System.out.println("El espacio " + numeroEspacio + " ya está ocupado");
        }
    }
}

class Tarifa {
    String tipoVehículo;
    double costoPorHora;

    public Tarifa (String tipoVehículo, double costoPorHora) {
        this.tipoVehículo = tipoVehículo;
        this.costoPorHora = costoPorHora;
    }
}
```



```

    }
    public double calcularCosto (int horas) {
    return costoPorHora * horas;
    }
    }

```

```

class Empleado {
    String nombre;
    int idEmpleado;

```

```

    public Empleado (String nombre, int idEmpleado) {
        this.nombre = nombre;
        this.idEmpleado = idEmpleado;
    }

```

```

    public void registrarEntrada (Vehiculo vehiculo) {
        System.out.println("Empleado " + nombre + " registro la entrada del vehiculo con placa " + vehiculo.placa);
    }
}

```

```

class Ticket {
    int numeroTicket;
    String horaSalida;
    double montoTotal;

```

```

    public Ticket (int numeroTicket, String horaSalida, double montoTotal) {
        this.numeroTicket = numeroTicket;
        this.horaSalida = horaSalida;
        this.montoTotal = montoTotal;
    }

```

```

    public void mostrarTicket () {
        System.out.println("Ticket# " + numeroTicket + " Hora de Salida " + horaSalida + " Monto a pagar $ " + montoTotal);
    }
}

```

```

public class Parquero {

```

```

    public static void main (String[] args) {

```

```

        Vehiculo vehiculo1 = new Vehiculo (placa: "ABC123", Tipo "Auto", horaEntrada: "10:00 AM");

```

```

        Vehiculo vehiculo2 = new Vehiculo (placa: "XYZ456", Tipo "Moto", horaEntrada: "11:00 AM");

```

```

        EspacioEstacionamiento espacio1 = new EspacioEstacionamiento (numero espacio: 1);

```

```

        EspacioEstacionamiento espacio2 = new EspacioEstacionamiento (numero espacio: 2);

```

```

        Tarifa tarifaMoto = new Tarifa (TipoVehiculo "Moto", costoPorHora: 3.0);
    }
}

```



Empleado empleado1 = new Empleado (nombre "Juan" idEmpleado 101);

empleado1. registrar Entrada (vehiculo 1);

espacio 1. asignar Vehiculo (vehiculo 1);

empleado1. registrar Entrada (vehiculo 2);

espacio 2. asignar vehiculo (vehiculo 2);

Tarifa ↴

Ticket ticketAuto = new Ticket (numeroTicket: 1, horaSalida: "12:00 PM", calcular costo (2));

Ticket ticketMoto = new Ticket (numeroTicket: 2, horaSalida: "12:30 PM", tarifaMoto. calcular costo (horas 4));

vehiculo1. Mostrar Detalles ();

ticket Auto. mostrar ticket ();

vehiculo 2. mostrar Detalles ();

ticket Moto. mostrar Ticket ();

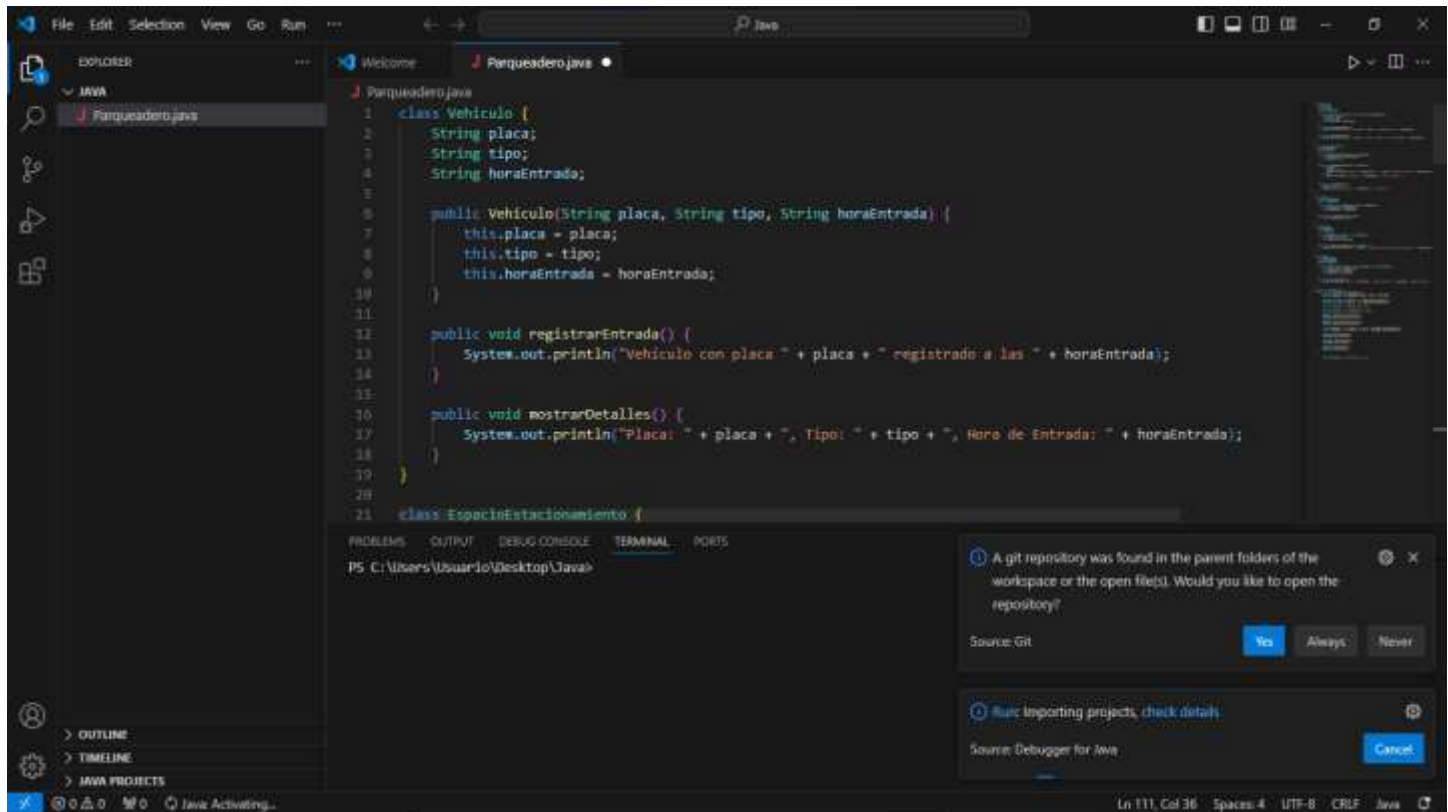
espacio 1. liberar Espacio ();

espacio 2. liberar Espacio ();

Tarifa tarifaAuto = new Tarifa (tipoVehiculo "Auto", costoPorHora: 5.0);



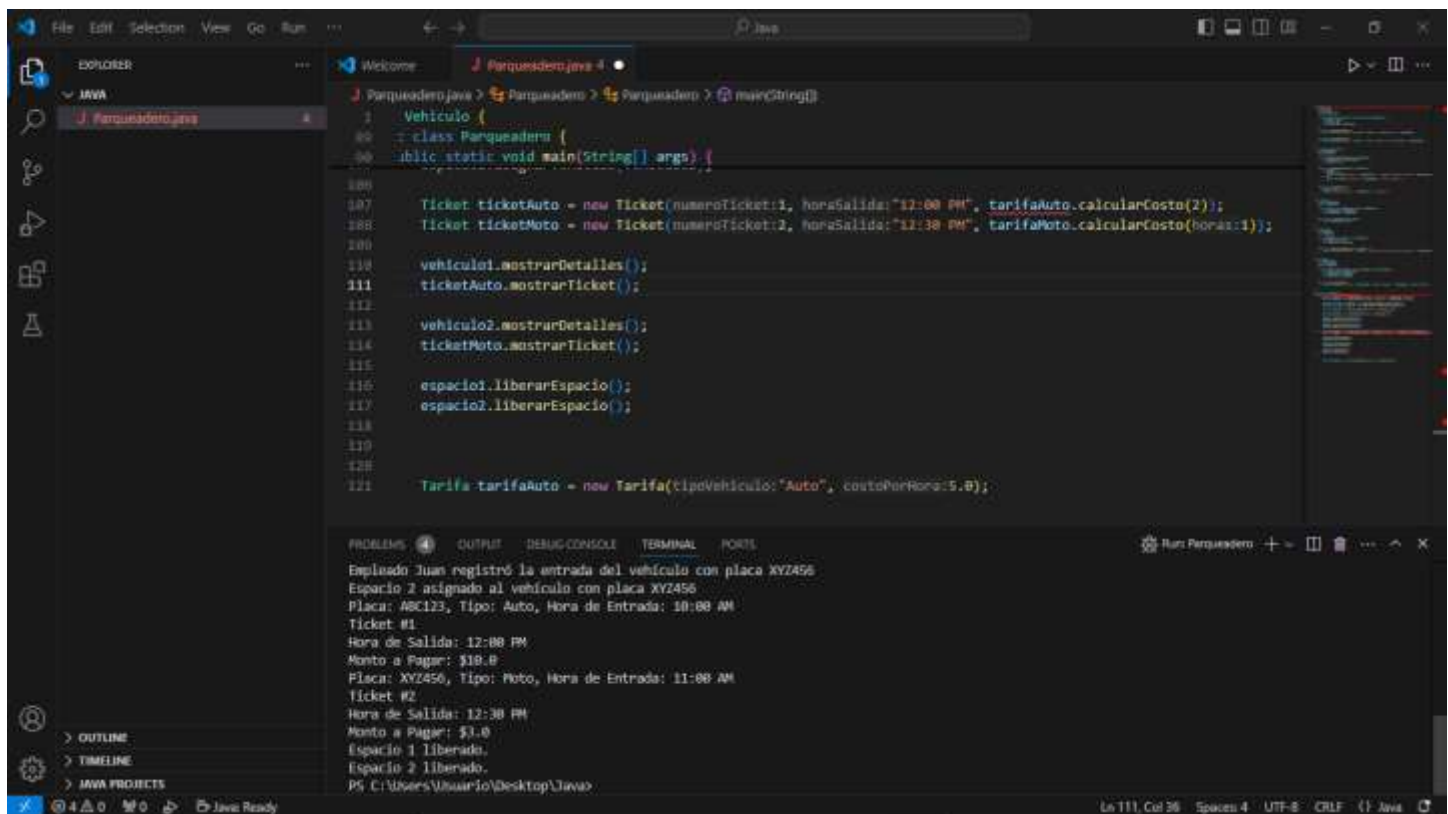
Evidencias, programado en Visual Studio Code



The screenshot shows the Visual Studio Code editor with the file `Parquadero.java` open. The code defines two classes: `Vehiculo` and `EspacioEstacionamiento`.

```
1 class Vehiculo {
2     String placa;
3     String tipo;
4     String horaEntrada;
5
6     public Vehiculo(String placa, String tipo, String horaEntrada) {
7         this.placa = placa;
8         this.tipo = tipo;
9         this.horaEntrada = horaEntrada;
10    }
11
12    public void registrarEntrada() {
13        System.out.println("Vehículo con placa " + placa + " registrado a las " + horaEntrada);
14    }
15
16    public void mostrarDetalles() {
17        System.out.println("Placa: " + placa + ", Tipo: " + tipo + ", Hora de Entrada: " + horaEntrada);
18    }
19 }
20
21 class EspacioEstacionamiento {
```

The bottom panel shows the `TERMINAL` tab with the command `PS C:\Users\Usuario\Desktop\Java>`. There are also two notifications on the right: "A git repository was found in the parent folders of the workspace or the open files. Would you like to open the repository?" and "Run Importing projects, check details".



The screenshot shows the Visual Studio Code editor with the file `Parquadero.java` open. The code defines the `main` method and creates instances of `Vehiculo` and `Tarifa`.

```
1 Vehiculo {
2     class Parquadero {
3         public static void main(String[] args) {
4
5             Ticket ticketAuto = new Ticket(numeroTicket:1, horaSalida:"12:00 PM", tarifaAuto.calcularCosto(2));
6             Ticket ticketMoto = new Ticket(numeroTicket:2, horaSalida:"12:30 PM", tarifaMoto.calcularCosto(horas:1));
7
8             vehiculo1.mostrarDetalles();
9             ticketAuto.mostrarTicket();
10
11             vehiculo2.mostrarDetalles();
12             ticketMoto.mostrarTicket();
13
14             espacio1.liberarEspacio();
15             espacio2.liberarEspacio();
16
17             Tarifa tarifaAuto = new Tarifa(tipoVehiculo:"Auto", costoPorHora:5.0);
```

The bottom panel shows the `TERMINAL` tab with the output of the program:

```
Empleado Juan registró la entrada del vehículo con placa XYZ456
Espacio 2 asignado al vehículo con placa XYZ456
Placa: ABC123, Tipo: Auto, Hora de Entrada: 10:00 AM
Ticket #1
Hora de Salida: 12:00 PM
Monto a Pagar: $10.0
Placa: XYZ456, Tipo: Moto, Hora de Entrada: 11:00 AM
Ticket #2
Hora de Salida: 12:30 PM
Monto a Pagar: $1.0
Espacio 1 liberado.
Espacio 2 liberado.
PS C:\Users\Usuario\Desktop\Java>
```

Resumen.

Se desarrollo este sistema como parte de nuestra práctica de Programación Orientada a Objetos. La idea es simular un sistema de estacionamiento donde se registran los vehículos, se asignan los espacios disponibles y se monitorea el tiempo de estacionamiento para ayudar a gestionar mejor el espacio y el tiempo de estacionamiento y a la vez veríamos los precios.

Bibliotecas:

Usamos la clase ArrayList de java.util para manejar las colecciones de vehículos y los espacios de estacionamiento. Elegimos esta clase porque es bastante flexible y nos permite agregar o quitar vehículos fácilmente sin tener que preocuparnos por el tamaño.

Descripción del Proyecto:

El sistema está hecho en Java y usa ArrayList para guardar tanto los vehículos como los espacios del parqueadero. Organizamos el proyecto en varias clases que representan a las entidades principales: Vehiculo, EspacioEstacionamiento, Empleado, Tarifa y Parqueadero. Cada clase tiene su propio rol, y se relacionan entre sí: por ejemplo, un Vehiculo puede estar asignado a un EspacioEstacionamiento y, con base en eso, se calcula el costo de estacionamiento con la clase Tarifa.

Conclusión:

Fue clave para entender cómo organizar el código en clases, cómo asociarlas entre sí y cómo aprovechar las colecciones dinámicas como ArrayList. También aprendimos a gestionar las relaciones entre las entidades del sistema. Aunque el sistema ya está funcionando bien, se puede mejorar y expandir agregando cosas como reportes o mejorando la interfaz. En resumen, este proyecto fue una excelente manera de aplicar lo que vimos en clase y de entender cómo estructurar sistemas más complejos.